NATIONAL

OS9

NEWSLETTER

Editor : Gordon Bentzen
8 Odin Street,
SUNNYBANK Qld. 4109.
(07) 345-5141

OCTOBER 1988

NATIONAL OS9 USERS GROUP NEWSLETTER

EDITOR : Gordon Bentzen
HELPERS : Bob Devries and Don Berrie

SUPPORT : Brisbane OS9 Level 2 Users Group

OS9 is the shape of things to come!

Are any of you thinking of purchasing a Compatible machine in the near future? If you do you may be in for a big surprise! "Big Blue" has suddenly found that MESS-DOS is just that, and that the power of the now available processors has finally surpassed the operating system. Viola! Now to introduce OS/2. A new operating system that allows - wait for it - Multi Tasking (!!!) and guess what else - sharable devices and files. It seems that IBM have finally found that in order to fully utilise the power of the 80386 chip, they needed to develop their own version of UNIX. And now we're gonna tell 'em that we've had all this for years!

Seriously, though. The message is that we have here an operating system that is still as modern as tomorrow, but is affordable, and runs on an affordable machine. At least when we speak about the CoCo Version. I would be interested to have any feedback form those of you who are using, or know of anyone else who is using other versions of OS9 on machines other than the CoCo. It would really help in the production of this Newsletter to know whether or not we have aimed in the right direction as far as content is concerned.

Why is OS9 so good? Well let's examine why we are apparently so happy with the system.

OS9 certainly has potential. There is no doubt about that. OS9 is fun to play with. In fact, to some of us that is the main attraction. It is a hackers delight. But I think that it goes deeper than that. The final value of the system will be judged upon the systems ability to provide an environment in which we can either do useful work, or provide a mechanism by which people can continue to learn about computing systems.

And OS9 has the ability to do all of this. We can play games, we can use it for business, we can use it for software development. Such a diversity of uses it what really makes the system so valuable.

This issue is the usual potpourri of goodies. We have included some comments and code from Bob Devries regarding the use of and some mistakes in the C windowing functions. Bob has really put some effort into this project of his, and he should be congratulated for his efforts.

As promised in an earlier Newsletter, this issue we are including the Basic09 source code for a CoCo disk maintenance utility by Don Berrie called ZAP. As the source code is rather long, we will be only be able to publish it in two parts. The second, and final instalment will appear in next months newsletter. As you can see, we have formatted it into two columns to try to fit as much as we can onto the limited paper space that we have. The formatting is on a per page basis. That is, the right column on each page follows after the left column, and thence to the following page.

We have also included a review of a commercial OS9 game, which is sold by Tandy. This review has been written by Nickolas Marentes, the author of a number of commercial programmes.

In the next issue, we plan to have some notes on the topic of communication, particularly in relation to the use of a terminal for the CoCo. The ability to use your MS-DOS machine as a terminal running under OS9 on your CoCo will really use it much more effectively than it can be used running under its own operating system.

By now it should be a familiar plea. However, we cannot go on indefinitely producing this newsletter by using the efforts of just a few people. This is a NEWSLETTER and that means that we need your input. Without it the amount and the quality of the sheet must decline. So come on, share that small snippet. Send us that piece of code. Let's hear from some non-CoCo users about their machines.

Multi-Vue and the C Programming language.

Boy, ain't we got fun! Here I was, a raw beginner at the art of C programming, and I thought I'd look at the whys and wherefores of using the C libraries that came with the Multi-Vue package. Well, to start off with, I thought that by following the examples in the manual, I couldn't go wrong. However, not only was I wrong, I was also led completely astray. Let me give you some examples of how far wrong you can get. The first thing I thought I'd try was to use some of the CGFX. LIB calls as documented in the manual. Did I come unstuck! The major reason for my sticky situation was that the manual mis-spelled several of the function names. For example, _ss_umbar is incorrect, it has more than 8 characters which is not allowed in our version of C. The correct spelling is _ss_ubar. When I found that the linker was spitting that one out as 'unresolved external', I decided that I should do a dump of the new CGFX library, and see what exactly was in it.

You use RDUMP for that, right? Wrong! It seems that the new library is set up differently so that the new linker RLINK can use them and the setup is different and RDUMP comes back with an error message to the effect that the file is not ROF. I ended up using a modified version of the programme 'LIB' from the user group library disks to read it. This gave me a file called 'modlist' which contained the names of all the cgfx functions, and their correct spellings of course. Please note that the C compiler is case sensitive.

Well, after trying several of the simpler graphics functions, I thought I'd try something a little more pretentious, and try a framed window with pull-down menus a-la multi-vue. I read through the manual again, and started to write my little programme. All I wanted at first was to create the framed window on an existing graphics screen, but when I tried to compile it, the C compiler told me of errors in the file 'stdmenu.h'. After examining it and referring again to the C compiler manual, I found that the compiler does not allow macros to be longer than one line. The 'stdmenu.h' file is full of them!

I decided to do my own thing inside my programme. Using the mouse drivers was also an experience in that while the information in the multi-vue manual is more or less correct, it leaves out some very important information and omits to mention one necessary function call to _ss_gip without which the mouse won't work at all. So after fighting with it for a while I conferred with Don Berrie on how he did his mouse drivers in Basic09 and there I found the missing call. After that session I worked out how to get the OS9 kernel to steer the graphics cursor around the screen with minimum of programming overhead. Now without further ado, here's the listing of the programme I have at this stage. Of course it was just a try out programme, and does not do anything at all except tell you what you selected.

```
#include <stdio.h>
#include <mouse.h>
#include <buffs.h>
#include <wind.h>

#define TRUE 1
#define FALSE 0
#define UPDATE 3
#define TIMOUT 10
#define FOLLOW 1
#define MOUSSIG 10
#define TNDY_MN {'T','a','n','d','y','\0'},MN_TNDY,10,9,1,0,0,_tanitms
#define FILE_MN {'F','i','l','e','s','\0'}, MN_FILE,10,7,1,0,0,_filitms
#define EDIT_MN {'E','d','i','t','\0'}, MN_EDIT,6,6,1,0,0,_edtitms
#define WXMIN 40
#define WYMIN 24
#define RESERVED {'\0','\0','\0','\0','\0','\0','\0'}
#define RES5 {'\0','\0','\0','\0','\0'}
```

```
MIDSCR _tanitms[] = {
    {{'C','a','l','c','\0'},1,RES5},
    {{'C','l','o','c','k','\0'},1,RES5},

{{'C','a','l','e','n','d','a','r','\0'},1,RES5},
    {{'C','o','n','t','r','o','l','\0'},1,RES5},
    {{'P','r','i','n','t','e','r','\0'},1,RES5},
    {{'P','o','r','t','\0'},1,RES5},
    {{'H','e','l','p','\0'},1,RES5},
    {{'S','h','e','l','l','\0'},1,RES5},
    {{'C','l','i','p','b','o','a',
'r','d','\0'},1,RES5}
};

MIDSCR _filitms[]  = {
    {{'N','e','w','\0'},1,RES5},
    {{'O','p','e','n','\0'},1,RES5},
    {{'S','a','v','e','\0'},1,RES5},
    {{'S','a','v','e',' ','A','s','.','.','.','.',
'\0'},1,RES5},
    {{'A','b','a','n','d','o','n','\0'},1,RES5},
    {{'P','r','i','n','t','\0'},1,RES5},
    {{'Q','u','i','t','\0'},1,RES5}
};

MIDSCR _edtitms[] = {
    {{'U','n','d','o','\0'},1,RES5},
    {{'C','u','t','\0'},1,RES5},
    {{'C','o','p','y','\0'},1,RES5},
    {{'P','a','s','t','e','\0'},1,RES5},
    {{'C','l','e','a','r','\0'},1,RES5},
    {{'S','h','o','w','\0'},1,RES5}
};

MNDSCR menus[] = {
       {TNDY_MN},
       {FILE_MN},
       {EDIT_MN}
       };


WNDSCR windat = {
       {'A','p','p','l','i','c','a','t','i','o',
'n','\0'},
       sizeof menus / sizeof menus[0],
       WXMIN,
       WYMIN,
       WINSYNC,
       RESERVED,
       menus
       };

MSRET msinfo;
int sigcode;

sighandler(sig)
int sig;
{
       sigcode = sig;
}

main()
{
       char path = 1;
       char ch;
       int itemno;
       int quit = FALSE;

       setbuf(stdin,NULL);
       setbuf(stdout,NULL);
       CurOff(path);
       if (_ss_wset(path,WT_FWIN,&windat)<0)
              exit(errno);
       intercept(sighandler);

       SetGC(path,GRP_PTR,PTR_ARR);
       _ss_gip(path,1,1,255,255);
       _ss_mous(path,UPDATE,TIMOUT,FOLLOW);
       _ss_msig(path,MOUSSIG);

       sigcode = 0;
       do {
            _ss_msig(path,MOUSSIG);
            if (sigcode == 0)
              tsleep(0);
            if (sigcode == MOUSSIG)
              {
              sigcode = 0;
              _gs_mous(path,&msinfo);
              if (msinfo.pt_stat == WR_CNTRL)
                {
                switch(_gs_msel(path,&itemno))
                  {
                  case MN_CLOS:
                    quit = TRUE;
                    break;
                  case MN_TNDY:
                    do_tandy_menu(itemno);
                    break;
                  case MN_FILE:
                    do_file_menu(itemno);
                    break;
                  case MN_EDIT:
                    do_edit_menu(itemno);
                    break;
                  }
                printf("%c",12);
                }
              }
       } while (!quit);
```

```
        SetGC(path,0,0);
        _ss_rel(path);
        CurOn(path);
        _ss_wset(path,WT_NBOX,&windat);
}

do_tandy_menu(item)
int item;
{
    char ch;
    switch(item)
        {
        case 1:
           printf("%c%c%c Calc.",2,42,42);
           break;
        case 2:
           printf("%c%c%c Clock.",2,42,42);
           break;
        case 3:
           printf("%c%c%c Calendar.",2,42,42);
           break;
        case 4:
           printf("%c%c%c Control.",2,42,42);
           break;
        case 5:
           printf("%c%c%c Printer.",2,42,42);
           break;
        case 6:
           printf("%c%c%c Port.",2,42,42);
           break;
        case 7:
           printf("%c%c%c Help.",2,42,42);
           break;
        case 8:
           printf("%c%c%c Shell.",2,42,42);
           break;
        case 9:
           printf("%c%c%c Clipboard.",2,42,42);
           break;
        }
        printf("\n\n       Press any key...");
        ch = getchar();
        printf("%c",12);
}
do_file_menu(item)
int item;
{
     char ch;
     switch (item)
          {
          case 1:
             printf("%c%c%c New.",2,42,42);
             break;
          case 2:
             printf("%c%c%c Open.",2,42,42);
             break;
          case 3:
             printf("%c%c%c Save.",2,42,42);
             break;
          case 4:
             printf("%c%c%c Save As...",2,42,42);
             break;
          case 5:
             printf("%c%c%c Abandon",2,42,42);
             break;
          case 6:
             printf("%c%c%c Print.",2,42,42);
             break;
          case 7:
             printf("%c%c%c Quit.",2,42,42);
             break;
          }
          printf("\n\n    Press any key...");
          ch = getchar();
          printf("%c",12);
}

do_edit_menu(item)
int item;
{
    char ch;
    switch(item)
        {
        case 1:
           printf("%c%c%c Undo.",2,42,42);
           break;
        case 2:
           printf("%c%c%c Cut.",2,42,42);
           break;
        case 3:
           printf("%c%c%c Copy.",2,42,42);
           break;
        case 4:
           printf("%c%c%c Paste.",2,42,42);
           break;
        case 5:
           printf("%c%c%c Clear.",2,42,42);
           break;
        case 6:
           printf("%c%c%c Show.",2,42,42);
           break;
        }
        printf("\n\n    Press any key...");
        ch = getchar();
        printf("%c",12);
}
```

ZAP

The following is the Basic09 source code for one of my favourite home-baked programmes, an OS9 Level 2 disk zapper. For those purists amongst you, I apologise for the spaghetti. However, by way of apology, I have to say that this project just grew, and grew, and grew! Consequently, it was written without much planning or attention to sensibilities. To have such things as line numbers etc. is archaic. However what you see is what you get. While the code was written using the data in the CoCo OS9 Level 2 Manual as reference material, I hope that it will function on any OS9 system. I will provide a complete description of the programme in the next newsletter.

Don Berrie.

```
    PROCEDURE zap
    BASE 0
    DIM PAGE,PATH,wpath,fgnd,bgnd,bord:BYTE
    DIM maxblock,ident,x,y,x1,y1:INTEGER
    DIM title:STRING[40]
    DIM CHOICE:STRING[25]
    DIM outstr:STRING[18]
    DIM NAME:STRING[4]
    DIM DTA,NEWBYTE:BYTE
    DIM COUNT,METER,BLKNO:REAL
    DIM OFFSET,iblkno:INTEGER
    DIM secdat(256):BYTE
    DIM keypress:STRING[1]
    DIM flag:BOOLEAN
    flag=FALSE
    fgnd=0
    bgnd=1
    bord=1
    OPEN #wpath,"/w":UPDATE
    RUN gfx2(wpath,"DWSET",2,0,0,80,24,fgnd,
bgnd,bord)
    RUN gfx2(wpath,"SELECT")
  1 fgnd=4
    bgnd=5
    RUN gfx2(wpath,"OWSET",1,0,0,80,24,fgnd,bgnd)
    RUN gfx2(wpath,"OWSET",0,1,0,79,24,fgnd,bgnd)
    x=0 \y=0 \x1=80 \y1=24
    outstr=""
    PAGE=0
    PRINT #wpath,CHR$($0C)
    RUN winopen(wpath,x,y,x1,y1)
    PRINT #wpath,
    PRINT #wpath," ";
    title="**   DISK MAINTENANCE UTILITY   **"
    RUN header(wpath,title)
    PRINT #wpath, \ PRINT #wpath,
    PRINT #wpath,"         (c) 1988 D.A.Berrie"
    PRINT #wpath,
    PRINT #wpath,"        Version 1.10 88/08/07"
    PRINT #wpath,
    PRINT #wpath,"## USE WITH CARE - PERMANENT
CHANGES TO DISK   STRUCTURE CAN BE MADE ##"
    PRINT #wpath,
    RUN getdev(wpath,NAME,PATH,secdat,maxblock,
ident)
    PRINT #wpath,
    RUN getsec(wpath,maxblock,BLKNO)
    ON ERROR GOTO 100
 10 METER=BLKNO*256
    iblkno=BLKNO
    COUNT=0
    SEEK #PATH,1
    GET #PATH,secdat
    IF secdat(14)*256+secdat(15)<>ident THEN
      x=6 \x1=40 \y=8 \y1=8
      RUN winopen(wpath,x,y,x1,y1)
      PRINT #wpath
      PRINT #wpath," WARNING: DISK HAS BEEN
CHANGED! "
      PRINT #wpath,"    Press any key to continue
 ";
      GET #wpath,keypress
      RUN winclose(wpath)
      ENDIF
    SEEK #PATH,METER
    GET #PATH,secdat
    CLOSE #PATH
 11 RUN scn(wpath,NAME,iblkno,secdat)
    IF flag=TRUE THEN
      keypress="A"
      flag=FALSE
      GOTO 21
      ENDIF
 20 fgnd=0
    bgnd=1
    RUN gfx2(wpath,"OWSET",1,6,22,32,1,fgnd,bgnd)
    RUN gfx2(wpath,"CUROFF")
    PRINT #wpath,"  A C D E H-help M N P Q <= =>
 ";
    GET #wpath,keypress
```

```
    RUN gfx2(wpath,"CURON")
    RUN gfx2(wpath,"OWEND")
 21 CHOICE=keypress
    IF CHOICE="D" OR CHOICE="d" THEN
      x=5 \x1=30 \y=10 \y1=6
      RUN winopen(wpath,x,y,x1,y1)
      RUN getdev(wpath,NAME,PATH,secdat,maxblock,
ident)
      RUN getsec(wpath,maxblock,BLKNO)
      GOTO 10
      ENDIF
    IF CHOICE="C" OR CHOICE="c" THEN
      x=40 \x1=30 \y=1 \y1=22
      RUN winopen(wpath,x,y,x1,y1)
      RUN calc(wpath)
      RUN winclose(wpath)
      GOTO 20
      ENDIF
    IF CHOICE="Q" OR CHOICE="q" THEN
      PRINT #wpath,CHR$($0C)
      RUN gfx2(1,"SELECT")
      RUN gfx2(wpath,"DWEND")
      CLOSE #wpath
      END
      ENDIF
    IF CHOICE="A" OR CHOICE="a" THEN
      x=40 \x1=30 \y=1 \y1=23
      RUN winopen(wpath,x,y,x1,y1)
      RUN ascii(wpath,secdat,keypress)
      RUN winclose(wpath)
      IF keypress=CHR$(8) OR keypress=CHR$(9) THEN

        flag=TRUE
        ENDIF
      GOTO 21
      ENDIF
    IF CHOICE="M" OR CHOICE="m" THEN
      RUN change(wpath,iblkno,secdat,PATH,NAME)
      GOTO 20
      ENDIF
    IF CHOICE="E" OR CHOICE="e" THEN
      x=0 \y=0 \x1=80 \y1=24
      RUN winopen(wpath,x,y,x1,y1)
      RUN directory(wpath)
      RUN winclose(wpath)
      GOTO 20
      ENDIF
    IF CHOICE="H" OR CHOICE="h" THEN
      x=40 \y=1 \x1=30 \y1=22
      RUN winopen(wpath,x,y,x1,y1)
      RUN helpmess(wpath,keypress)
      RUN winclose(wpath)
      GOTO 21
      ENDIF
    IF CHOICE="N" OR CHOICE="n" THEN
      OPEN #PATH,NAME
      x=20 \y=12 \x1=40 \y1=5
      RUN winopen(wpath,x,y,x1,y1)
      RUN getsec(wpath,maxblock,BLKNO)
      GOTO 10
      ENDIF
    IF CHOICE=CHR$(9) THEN
      OPEN #PATH,NAME
      BLKNO=BLKNO+1
      IF BLKNO>maxblock THEN
        BLKNO=0
        GOTO 10
        ENDIF
      GOTO 10
      ENDIF
    IF CHOICE=CHR$(8) THEN
      OPEN #PATH,NAME
      BLKNO=BLKNO-1
      IF BLKNO<0 THEN
        BLKNO=maxblock
        GOTO 10
        ENDIF
      GOTO 10
      ENDIF
    IF CHOICE="P" OR CHOICE="p" THEN
 25   x=25 \y=10 \x1=30 \y1=5
      COUNT=0
      RUN winopen(wpath,x,y,x1,y1)
      PRINT #wpath,"    Printing Sector ";
      ON ERROR GOTO 30
      OPEN #PATH,"/p"
      PRINT #PATH,"DEVICE : "; LEFT$(NAME,
LEN(NAME)-1); "   ";     "SECTOR : $";
      PRINT #PATH USING "h4>",iblkno
      PRINT #PATH
      PRINT #PATH,"Rel  0 1 2 3 4 5 6 7 8 9 A B C
D E F       0 2   4   6 8 A C E"
      PRINT #PATH,"Addr     ------------------
-----------------------------------"
    FOR i=0 TO 255
      IF COUNT=0 THEN
        PRINT #PATH USING "H2>",PAGE;
        PAGE=PAGE+1
        PRINT #PATH,"   ";
        ENDIF
      PRINT #PATH USING "h2>",secdat(i);
      COUNT=COUNT+1
      IF secdat(i)<$21 OR secdat(i)>$7A THEN
        outstr=outstr+"."
        ELSE
        outstr=outstr+CHR$(secdat(i))
        ENDIF
      IF COUNT=16 THEN
        PRINT #PATH,"       "; outstr
        outstr=""
```

```
        COUNT=0
        ENDIF
      NEXT i
    PRINT #PATH,CHR$($0C)
    CLOSE #PATH
    PAGE=0
    RUN winclose(wpath)
    ENDIF
  GOTO 20
  END
 30 errnum=ERR
  RUN winclose(wpath)
  IF errnum=246 THEN
    x=25 \y=10 \x1=40 \y1=5
    RUN winopen(wpath,x,y,x1,y1)
    PRINT #wpath,"PLACE PRINTER ONLINE - PRESS A
KEY";
    GET #wpath,keypress
    keypress=""
    RUN winclose(wpath)
    GOTO 25
    ENDIF
100 RUN closerr(wpath)
  END

  PROCEDURE helpmess
  ON ERROR GOTO 100
  PARAM wpath:BYTE
  PARAM keypress:STRING[1]
  DIM title:STRING[40]
  PRINT #wpath,"   ";
  title="DISK MANAGEMENT HELP"
  RUN header(wpath,title)
  PRINT #wpath,
  PRINT #wpath,
  PRINT #wpath,"A - Produces Ascii dump"
  PRINT #wpath,"C - Calculator (Hex.)"
  PRINT #wpath,"D - Change RBF Device"
  PRINT #wpath,"E - Extended directory"
  PRINT #wpath,"H - Produces this list"
  PRINT #wpath,"M - Modify current sector"
  PRINT #wpath,"N - Change sector number"
  PRINT #wpath,"P - Hardcopy of sector"
  PRINT #wpath,"Q - Quit program"
  PRINT #wpath,"<= (left arrow) - change"
  PRINT #wpath,"   to previous sector"
  PRINT #wpath,"=> (right arrow) - change"
  PRINT #wpath,"   to next sector"
  PRINT #wpath,
  PRINT #wpath,"Select Any Key";
  GET #wpath,keypress
  END
100 RUN closerr(wpath)
  END

  PROCEDURE scn
  ON ERROR GOTO 100
  BASE 0
  PARAM wpath:BYTE
  PARAM NAME:STRING[4]
  PARAM iblkno:INTEGER
  PARAM secdat(256):BYTE
  DIM COUNT:REAL
  DIM i:INTEGER
  DIM PAGE:BYTE
  DIM outstr:STRING[16]
  PRINT #wpath,CHR$($0C)
  PAGE=0 \outstr=""
  COUNT=0
  PRINT #wpath,"DEVICE : "; LEFT$(NAME,
LEN(NAME)-1); "  ";     "SECTOR : $";
  PRINT #wpath USING "H4>",iblkno
  PRINT #wpath,
  PRINT #wpath,"Rel  0 1 2 3 4 5 6 7 8 9 A B C D
E F "
  PRINT #wpath,"Addr --------------------------
-----"
  FOR i=0 TO 255
    IF COUNT=0 THEN
      PRINT #wpath USING "h2>",PAGE;
      PAGE=PAGE+1
      PRINT #wpath,"   ";
      ENDIF
    PRINT #wpath USING "h2>",secdat(i);
    COUNT=COUNT+1
    IF secdat(i)<$21 OR secdat(i)>$7A THEN
      outstr=outstr+"."
      ELSE
      outstr=outstr+CHR$(secdat(i))
      ENDIF
    IF COUNT=16 THEN
      PRINT #wpath
      COUNT=0
      ENDIF
    NEXT i
  PRINT #wpath
  END
100 RUN closerr(wpath)
  END

  PROCEDURE winopen
  ON ERROR GOTO 100
  PARAM wpath:BYTE
  PARAM x,y,x1,y1:INTEGER
  RUN gfx2(wpath,"owset",1,x,y,x1,y1,1,2)
  RUN gfx2(wpath,"owset",1,x+1,y+1,x1-2,
y1-2,0,1)
  RUN gfx2(wpath,"owset",1,x+2,y+2,x1-4,
y1-4,0,1)
  x=0 \y=0 \x1=0 \y1=0
```

```
    END
100 RUN closerr(wpath)
    END

    PROCEDURE winclose
    PARAM wpath:BYTE
    ON ERROR GOTO 100
    RUN gfx2(wpath,"owend")
    RUN gfx2(wpath,"owend")
    RUN gfx2(wpath,"owend")
    END
100 RUN closerr(wpath)
    END

    PROCEDURE ascii
    ON ERROR GOTO 100
    BASE 0
    PARAM wpath:BYTE
    PARAM secdat(256):BYTE
    PARAM keypress:STRING[1]
    DIM i,count:INTEGER
    DIM page:BYTE
    DIM outstr:STRING[16]
    outstr=""
    page=0
    count=0
    PRINT #wpath,"      0 2 4 6 8 A C E"
    PRINT #wpath,"      ----------------"
    FOR i=0 TO 255
      IF count=0 THEN
        PRINT #wpath USING "h2)",page;
        page=page+1
        PRINT #wpath,"    ";
        ENDIF
      IF secdat(i)<$20 OR secdat(i)>$7A THEN
        outstr=outstr+"."
        ELSE
        outstr=outstr+CHR$(secdat(i))
        ENDIF
      count=count+1
      IF count=16 THEN
        PRINT #wpath,outstr
        outstr=""
        count=0
        ENDIF
      NEXT i
    PRINT #wpath,"Select A Key";
    GET #wpath,keypress
    END
100 RUN closerr(wpath)
    END

    PROCEDURE change
    ON ERROR GOTO 100
    BASE 0
    PARAM wpath:BYTE
    PARAM iblkno:INTEGER
    PARAM secdat(256):BYTE
    PARAM PATH:BYTE
    PARAM NAME:STRING[4]
    DIM x,y,x1,y1:INTEGER
    DIM key:STRING[1]
    DIM hex:STRING[2]
    DIM a,b,c,d,ihex:INTEGER
    DIM flag:INTEGER
    DIM first:BOOLEAN
    flag=0
    a=5 \b=5 \c=5 \d=5
    RUN gfx2(wpath,"curoff")
    RUN gfx2(wpath,"CURXY",5,22)
    RUN gfx2(wpath,"COLOR",0,1)
    PRINT #wpath," 2-HEX-BYTES H-HELP W Q arrows
";
    RUN gfx2(wpath,"COLOR",4,5)
 10 c=a \d=b
    RUN swopen(wpath,a,b,secdat)
    RUN gfx2(wpath,"owset",1,78,23,1,1,5,5)
    GET #wpath,key
    RUN gfx2(wpath,"owend")
 11 hex=""
    first=FALSE
    IF key=CHR$(10) THEN
      b=b+1
      IF b=21 THEN
        b=5
        ENDIF
      RUN swclose(wpath,a,b,c,d,secdat)
      GOTO 10
      ENDIF
    IF key=CHR$(12) THEN
      b=b-1
      IF b=4 THEN
        b=20
        ENDIF
      RUN swclose(wpath,a,b,c,d,secdat)
      GOTO 10
      ENDIF
    IF key=CHR$(9) THEN
      a=a+2
      IF a=37 THEN
        a=5
        b=b+1
        IF b=21 THEN
          b=5
          ENDIF
        ENDIF
      RUN swclose(wpath,a,b,c,d,secdat)
      GOTO 10
      ENDIF
    IF key=CHR$(8) THEN
```

```
      a=a-2
      IF a=3 THEN
        a=35
        b=b-1
        IF b=4 THEN
          b=20
          ENDIF
        ENDIF
      RUN swclose(wpath,a,b,c,d,secdat)
      GOTO 10
      ENDIF
 15 IF ASC(key)>47 AND ASC(key)<58 OR ASC(key)>64
AND ASC(key)<71 OR ASC(key)>96 AND ASC(key)<103
THEN
      hex=hex+key
      IF first=TRUE THEN
        GOTO 16
        ENDIF
      first=TRUE
      key=""
      GET #wpath,key
      GOTO 15
 16   secdat((a-5)/2+(b-5)*16)=VAL("$"+hex)
      flag=1
      RUN swopen(wpath,a,b,secdat)
      a=a+2
      IF a=37 THEN
        a=5
        b=b+1
        IF b=21 THEN
          b=5
          ENDIF
        ENDIF
      RUN swclose(wpath,a,b,c,d,secdat)
      GOTO 10
      ENDIF
    IF key="W" OR key="w" THEN
      RUN swclose(wpath,a,b,c,d,secdat)
      RUN gfx2(wpath,"curon")
      RUN modify(wpath,secdat,iblkno,
PATH,NAME,flag)
      END
      ENDIF
    IF key="Q" OR key="q" THEN
      GOTO 20
      ENDIF
  IF key="H" OR key="h" THEN
    RUN swclose(wpath,a,b,c,d,secdat)
    x=45 \y=1 \x1=30 \y1=22
    RUN winopen(wpath,x,y,x1,y1)
    RUN helpmess2(wpath,key)
    RUN winclose(wpath)
    GOTO 11
    ENDIF
  key=""
    GOTO 10
 20 key=""
    RUN swclose(wpath,a,b,c,d,secdat)
    IF flag=1 THEN
      x=10 \x1=60 \y=8 \y1=8
      RUN winopen(wpath,x,y,x1,y1)
      PRINT #wpath,"    NO CHANGES SAVED TO DISK
"
      PRINT #wpath,
      PRINT #wpath,"   Press Any Key to Continue"
      GET #wpath,key
      RUN winclose(wpath)
      ENDIF
    RUN gfx2(wpath,"CURXY",5,22)
    PRINT #wpath,"
";
    RUN gfx2(wpath,"curon")
    END
100 RUN closerr(wpath)
    END

    PROCEDURE swopen
    ON ERROR GOTO 100
    BASE 0
    PARAM wpath:BYTE
    PARAM a,b:INTEGER
    PARAM secdat(256):BYTE
    RUN gfx2(wpath,"curxy",a,b)
    RUN gfx2(wpath,"revon")
    PRINT #wpath USING "h2>",secdat((a-5)
/2+(b-5)*16);
    RUN gfx2(wpath,"curxy",a,b)
    END
100 RUN closerr(wpath)
    END

    PROCEDURE swclose
    ON ERROR GOTO 100
    BASE 0
    PARAM wpath:BYTE
    PARAM a,b,c,d:INTEGER
    PARAM secdat(256):BYTE
    RUN gfx2(wpath,"revoff")
    RUN gfx2(wpath,"curxy",c,d)
    PRINT #wpath USING "h2>",secdat((c-5)
/2+(d-5)*16);
    RUN gfx2(wpath,"curxy",a,b)
    END
100 RUN closerr(wpath)
```

##
K O R O N I S R I F T
S O F T W A R E R E V I E W
##

By Nickolas Marentes (CoCo3 Commercial Programmer)

GAME SCENARIO:

You are a 'Techno-Scavenger', one who makes a living searching for abandoned technological systems. From galaxy to galaxy you roam collecting technological "junk" for resale. Life's tuff!
Suddenly, your instruments spring to life like they have never done so before. Radiation flux levels in the ten thousand range! Closer examination reveals that you have stumbled across the fabled "Koronis Rift". An ancient test ground for the most powerful weapons.
On descending into the rifts, you drive your Surface Rover across the planet surface, tracking down abandoned "Hulks". Once found, you send you Repo-Tech droid to loot it and return with any useful systems. You must analyze each module, working out what each module is, how much energy it has, how much it's worth and determine if you can put it to use in your own Surface Rover for increased capabilities.
Beware though, the planet is patrolled by Guardian Saucers which you must destroy or evade.

GAME PACKAGING:

The game comes very well presented in a professionally presented, colour box. Inside is a small well written booklet, a command card and a disk. Koronis Rift has been available for the Atari 400/800 and Commodore 64 computers before finally being released for the CoCo3. The documentation is designed for each computer with the command card being specific for the host system.

PROGRAM DEVELOPMENT:

The program is brought out by EPYX, a large U.S. software house who has mainly been supporting the Commodore 64 computer. The program is actually developed by a team called LUCASFILM LTD. which is the home computer software division of George Lucas's (Star Wars fame) special effects and film company. So, as you can see, Koronis Rift is no "backyard job". The CoCo3 version programmers are Edwin Rosenzweig and Ken Rogoway. The program runs under the CoCo3's OS-9 Level 2 system.

POSITIVE POINTS:

Good use of the OS-9 environment. Good use of the CoCo3's graphics (great title page!) making use of colour shades for added depth. Great game scenario of which much of the game terminology are trademarks of Lucasfilm Ltd.

NEGATIVE POINTS:

Sound is a bit on the minus side, especially when compared with the Atari and Commodore versions but this is a speed limitation when running under OS-9 and the CoCo3's lack of a dedicated sound chip.

CLOSING COMMENTS:

Great program! The program is good value for money when one considers the professionalism of packaging, depth of script and development of special Fractal algorithms for the planet terrain. One very unfortunate thing

though, according to Tandy here in Australia, Koronis Rift is now a discontinued product! I for one am shocked that such good software, which hasn't had the chance to be marketed properly is being given the boot. I can think of other programs which Tandy are selling which I feel should be discontinued instead of this one (stay tuned for a future episode!). So, if you're planning on getting a new CoCo3 game, grab Koronis Rift before they completely disappear.

AVAILABLE: Tandy Electronics Stores
PRICE : $52.46 (Discontinued price)
REQUIRES : 128K CoCo3 + Disk Drive

ooOOOoo

The following is a list of system error messages which may be of use as an easily accessable reference :

OS9 ERROR MESSAGES

183 - Illegeal window type
184 - Window already defined
185 - Font not found
186 - Stack overflow
187 - Illegal argument
188 - unused
189 - Illegal Coordinates
190 - Internal integrity check
191 - Buffer size too small
192 - Illegal command
193 - Screen or Window table is full
194 - Bad/Undefined buffer number
195 - Illegal window definition
196 - Window undefined
197 - unused
198 - unused
199 - unused
200 - Path tabl full
201 - Illegal path number
202 - Interrupt poling table full
203 - Illegal mode
204 - Device table full
205 - Illegal module header
206 - Module directory full
207 - Memory full
208 - Illegal service request
209 - Module busy
210 - Boundary error
211 - End of file
212 - Returning non-allocated memory
213 - Non-existing segment
214 - No permission
215 - Bad path name
216 - Path name not found
217 - Segment list full
218 - File already exists
219 - Illegal block address
220 - Data carrier detect lost
221 - Module not found
223 - Suicide attempt
224 - Illegal process num,ber
226 - No children
227 - Illegal SWI code
228 - Process aborted
229 - Process table full
230 - Illegal parameter area
231 - Known module
232 - Incorrect module CRC
233 - Signal error
234 - Non-existent Module
235 - Bad Name
236 - Bad module header
237 - RAM full
238 - Unknown process ID
239 - No task number available
240 - Unit error
241 - Sector error
242 - Write protect
243 - CRC error
244 - Read error
245 - Write error
246 - Not ready
247 - Seek error
248 - Media full
249 - Wrong type
250 - Device ready
251 - Disk ID change
252 - Record is locked out
253 - Non-sharable file busy
254 - I/O deadlock errror